# Learning Database through Developing Database Web Applications

Ching-Yu Huang

***Abstract*—Databases play a very important role in the modern software system, especially in the big data age. It is the centralized repository for all kinds of data. Therefore, all students majoring in technology and computer science related fields should take database courses. However, learning about databases could be challenging without hands-on exercises. It is very important for students to be familiar with database commands and the associated clauses so they can write better database queries. Since the data resides in the database, it is hard for students to imagine the results, especially if there are a lot of records. Visualizing the intermediate results will help students to learn how to write better queries and troubleshoot. It is not practical to just use the command line to interact with the database server. Students must learn the industry standard 3-tier method to retrieve, insert, update and delete thousands of data from databases, and how to write web programs to do so. Designing an E-commerce platform should be the core of fundamental database curriculum - including a friendly graphic user interface for customers to sign up for accounts, search items, place orders and view histories. The instructors should not focus on teaching the database topics, but also show students how to manage the database server and user accounts in the classes. This paper proposes an integrated curriculum for learning about databases through developing database web applications.**

***Index Terms*—Apache, database web applications, PHP, MySQL, XAMPP.**

## I. Introduction

Before the first commercial Database Management System (DBMS) was introduced by Oracle in late 1970s [1], teaching Databases was mainly based on theoretical concepts. Even in late 1980s, most schools could not afford to buy the expensive DBMS, preventing students from attaining hands-on experience. After free and open source relational DBMS, such as MySQL [2], was made available to the public in the mid-1990s, students were able to utilize DB for projects. Although MySQL originally did not have many functions during its early developmental stages, performing basic SQL queries significantly helped students gain hands-on experience for data management.

Since the data is stored in the database, the ability to retrieve and display data is critical. Even though the DBMS provides both a command line interface and a Graphic User Interface (GUI) for users to retrieve, insert, update, and delete records, it is often difficult for non-computer science employees to write SQL statements through these interfaces to manually work on the DB. It is impossible for external customers to access the DB directly without any application interface. Therefore, developing DB applications to help users (staff and customers) access data is very important for scaling up any business. Since web-based applications are becoming essential for various industries, a web DB application should one of the main focuses for projects in any DB course.

In order for a computer language to communicate with the DBMS, specific drivers are required for each language. Different DBMS require users to install different DB drivers, which are usually provided by the vendors. In addition, many DB drivers are Operating System (OS) platform dependent, especially for C, C++, and other compiled languages. Scripting languages such as Perl and Python use a standard DB library across different platforms [3], [4]. Even through Java is a platform independent programming language, it still requires connecting packages which are OS dependent. Now, more and more languages are being introduced which better accommodate the integration between web and database servers. These languages, which are considered server-side, must be run on web servers. For example, Hypertext Preprocessor (PHP) [3] is one of the most popular languages for those working on web DB applications. Apache [4], a free open source web server, has a built-in module that allows users to integrate PHP with MySQL [5].

It is difficult for DB instructors to set up and manage both Web and DB servers and manage student accounts, while teaching DB courses at the same time. An integrated software package, XAMPP, was introduced in 2002 which helps combat this issue [6]. The package is cross platform (X), and it includes an Apache (A) web server and a MySQL (M) database, as well as Perl (P) and Python (P) compatibility. The latest XAMPP also has phpMyAdmin – a free and open source administration tool for MySQL and MariaDB – and Tomcat, a web server for Java Servlet. Additionally, XAMPP already has PHP built-in. It is available for Windows, iOS, and Linux systems and does not require powerful hardware. This allows students to easily install XAMPP on their laptops, and run an Apache web server and a MySQL server at the same time. Once students are familiar with the client-server model, it will be easier to pick up the 3-tier architecture that is often used in the real world, where each is typically stored on a separate machine.

This paper will utilize XAMPP as a part of the proposed curriculum and hands-on exercises. MySQL will be used as the database to learn SQL, phpMyAdmin [7] will be used for creating users and managing MySQL DBMS, and MySQL Workbench – a visual administration tool to manage MySQL DB – will be used as the GUI for writing SQL queries and viewing the results. PHP and Apache will be the languages used for developing web applications.

Manuscript received September 12, 2018; revised December 4, 2018.

Ching-Yu Huang is with School of Computer Science, Kean University, Union, New Jersey, USA (e-mail: chuang@kean.edu).

doi: 10.18178/ijiet.2019.9.4.1207

## II. Database Curriculum

Relational DB [8] are often adopted for teaching purposes because it still accounts for the majority of the DB job market; thus, they will be the focus of the curriculum. The proposed DB curriculum is based on 40 total hours - 32 hours of total lesson time (13 weeks and 2.5 hours per week) and 4 exams, with 2 hours allocated for each, including reviewing the answers. Students should have at least a basic level of programming knowledge with a good foundation in variables, loops, logic conditions, if statements, arrays, and functions. The curriculum will build on this prerequisite knowledge by focusing on basic DB knowledge and its application. Theoretical concepts such as relational calculus, and advanced topics such as triggers, transactions, concurrency control, query Processing & optimization, DBA, and data mining are not covered. The assignments include 2 homework assignments and one web DB application project, with two phases.

Instructors should prepare several tables with records and export the tables into SQL files for students to download. This way, students can import the tables into their local MySQL server and work on the same tables and dataset as that of the instructors. Consequently, everyone should obtain the same results when doing exercises with the same SQL statements. If students have any difficulties, instructors can quickly help the students. Students should do more exercises on basic SQL operations, and should not be limited to the examples shown in this paper, as this is not intended to be a textbook. The following subsections detail the weekly curriculum.

### A. Introduction of DB

The first week of lessons should include the following topics: class policy, the course schedule, an overview of the DB curriculum, basic DB concepts, and the role of DB in modern software systems. The fundamental knowledge of the client-server model and 3-tier architecture should be covered. Instructors should also help students install the XAMPP package, and make sure the MySQL and Apache web servers can be run successfully.

### B. BASIC SQL — Select, Where, Sorting

The 2nd week should cover basic concepts of tables, rows, columns, and data types. This week focus on the SQL SELECT command and its syntax, with AS, LIKE, DISTINCT, ORDER BY, AND, OR, NOT, and WHERE clauses. Students should be able to select specific records and columns with simple and compound conditions. In addition, one should cover how to sort the output. Homework 1 should be given in the 2nd week, and due at the end of the 6th week. It should mainly focus on the SELECT queries. As a result of this week's curriculum, students should be able to write the following SQL statements based on Table I, a sample table of staff members at a company.

1) The following SQL statement shows all columns, for all staff.

SELECT * FROM Staff;

2) The following SQL statement shows the salary and department for any staff named Mary.

SELECT salary, dept FROM Staff WHERE name='Mary';

3) The following SQL statement shows the unique department for any staff who has name contains the letter "e".

SELECT distinct dept from Staff WHERE name like '%e%';

4) The following SQL statement shows the names of all staff who are female and have a salary >= 40000.

SELECT name FROM Staff WHERE sex='F' and salary >= 40000;

5) The following SQL statement shows the names and salaries of all staff that work at IT department, with the output sorted from high to low (in salary).

SELECT name, salary FROM Staff WHERE dept='IT' ORDER BY salary DESC;

TABLE I: Staff Table

| sid | name | dept | sex | salary | ext |
|---|---|---|---|---|---|
| 101 | Mary | IT | F | 45000 | 121 |
| 102 | Smith | HR | M | 37000 | NULL |
| 103 | Tony | IT | M | 38000 | 115 |
| 104 | Sarah | PJ | F | 35000 | 311 |
| 105 | Mark | IT | M | 45000 | 153 |
| 106 | Alice | HR | F | 39000 | 433 |
| 107 | Ben | IT | F | NULL | NULL |

### C. Web DB Application

In the 3rd week, students should learn how to display the data from the DB on the browser after learning the basic SELECT statements. This should be done twofold: first, the instructor should quickly cover basic web design and how to take input and pass the values to PHP programs on the Apache web server. Then, students should learn how to use PHP to retrieve the data from the DB using the input values. The main goal of this paper is to focus on learning the basic DB queries and integrating the DB and Web. This means only basic web programming is required to show how the web and DB work together. Students should take web programming courses for learning how to design better web pages and more aesthetic GUI. Project 1 should be given in the 3rd week and should be due at the end of the 7th week. The details of the project are described in section 2.4.

#### 1) Input on web page (browser)

Hypertext Markup Language (HTML) is a markup language to create web pages and web applications using tags <>. Many HTML tags have attributes that can modify elements, specify tags' values, or perform specific actions. The browser will interpret the tags and attributes, with specific syntax, that will take input and display results. The HTML file should have the *.html extension and should begin with an <HTML> tag and end with an </HTML> tag. Fig. 1 shows the HTML source code of a simple web page that allows a user to enter input. The web page is shown in Fig. 2. Most HTML tags require end tags </> to match the begin tag <>.

The message written between <TITLE> and </TITLE> will be displayed as the title of the web page. The <BR> tag creates a new single line break. The <input> tag specifies an input field where the user can enter data. <input> elements are used within a <form> element to declare input controls which allow users to input data [9]. An input field can vary in many ways, depending on the type attribute. The first

<INPUT> tag allows users to enter text, and the 2nd <INPUT> tag is a submit type which will ask the browser to pass the values of <INPUT> tags, which are placed between <FORM> and </FORM> tags, to the program on the web server. More information about HTML tags can be referenced at w3schools.com [10].

```
<HTML>
<TITLE>Test</TITLE>
Web DB application: search department
<BR>
<FORM action='check.php'>
Department:
<INPUT type='text' name='keyword' method='GET'>
<INPUT type='submit' value='Check'>
</FORM>
</HTML>
```

Fig. 1. The HTML source code of test.html.


Fig. 2. The display on the browser from the HTML code shown in Fig. 1.

*2) Common gateway interface (CGI)*

A browser uses the CGI protocol to send the information to a web server [11]. When the input values and other information shown on the browser are passed to the web server using the <FORM> tag, the browser also tells the web server to call the specific program defined in the action attribute of the <FORM> tag. For example, the PHP program check.php, which is defined inside the <FORM> tag in Fig. 1, is called with the specific input. In order for this to work, the program should be able to be executed directly by the OS where the web server is running. Most of the CGI programs are nowadays written in scripting languages because it is easier to maintain. When the web server receives a message from the browser, it will trigger the OS to call the language interpreter to run the program.

There are two methods can be defined in the <FORM> to pass the information from the browser to the web server – GET and POST. The GET method will show the message string (name/value pairs) that is sent in the URL. Many search functions are implemented in the GET method so people can change keyword values in the URL. This will allow other programs to call the URL directly and obtain the results. The POST method delivers a message in the HTTP message body, which is not visible from the URL. If the message contains confidential information, such as a password or other information that should not be visible, it is better to use the POST method.

*3) Web CGI programming using PHP*

The PHP program that the programmer wishes to call should be written between the strings "<?php" and "?>". For more PHP reference materials, tutorials for PHP can be found at w3schools.com [11]. Once the CGI program is called, it will receive all the <input> tags' names and values. In PHP, special arrays $_POST[] and $_GET[] are used to get the values of the corresponding name in the array (note that the $ sign is used to indicate variables in PHP). For example, the following statement will assign the 'keyword' value of <INPUT> tag in Fig. 1 to a PHP variable $value.

*$value=$_GET['keyword'];*

*4) Access MySQL using PHP*

PHP has built-in functions to work with MySQL DB [12]. The basic steps are as follow: 1. Establish a connection to the DB on a server with a valid login and password. 2. Send the query string. 3. Receive the results and place them in a PHP array. 4. Check the number of rows in the array. 5. Fetch all the records in the result array. 6. Free the result array and close the connection. A sample program that follows this procedure is shown in Fig. 3. The program connects to the "test" database on the "localhost" server using the login "tester" and password "1234".

If the DB connection fails, the program can use *mysqli_connect_error()* to show the error message returned from the DB; the program should terminate immediately and display the error message. It will be a waste of the web server's resources if the login fails but the program continues to run the rest of the statements.

As shown in the following PHP statement, a variable $sql stores the query string which selects the names and salaries from the Staff table with condition dept='$value' (as assigned in section 2.3.3).

*$sql= "SELECT name, salary FROM Staff where dept='$value'";*

Please note that if the attribute is not a numeric type, the PHP variable $value should be surrounded with single quotes for logic comparisons, and numeric values should not be wrapped with single quotes for comparison. For example, to compare the salary column with the PHP variable, the correct comparison statement is *salary=$value* (to see if they are equal).

The function *mysqli_query()* will send the query string to the DB through the variable $conn. If the query is successfully executed by the DB server, the returned results are saved into the 2-D array $result. The function *mysqli_num_rows()* returns the number of records stored in the array. The function *mysqli_fetch_assoc()* will fetch a record each time from the 2-D array and the record is saved to a 1-D array $row. The while loop will read every record in the 2-D array. The values of name and salary can be retrieved from the 1-D array $row using the table column name "name" and "salary" as the keys. The results are displayed on the browser using the <TABLE> tag and its related tags: table row tag <TR>, table header tag <TH>, and table data/cell tag <TD>.

Fig. 4 shows the results of a search with the keyword "HR", displayed in a table with 2 columns (name and salary) and 3 rows. If the search does not yield any results, a warning message should be displayed so users know what happened. Fig. 5 shows the results if no record is found (i.e. if the search keyword is "XY"). For the 3-tier architecture commonly used with web DB applications, bugs and problems might arise at front-end, middle-end, or back-end. Therefore, it is important to keep users informed with meaningful messages about the running status.

```php
<?php
$value=$_GET['keyword'];
$server = "localhost";
$user = "tester";
$pass = "1234";
$db = "test";

$conn = mysqli_connect($server,$user,$pass,$db);
if (!$conn)
  die("Connection failed: " . mysqli_connect_error());

$sql = "SELECT name, salary FROM Staff where dept='$value'";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {
  echo "Records found for dept: $value\n";
  echo "<TABLE border=1>\n";
  echo "<TR><TH>Name<TH>Salary\n";
  while($row = mysqli_fetch_assoc($result)) {
    echo "<TR><TD>".$row['name']."<TD>".$row['salary']."\n";
  }
  echo "</TABLE>\n";
} else
  echo "No record is found for dept: $value\n";

mysqli_free_result($result);
mysqli_close($conn);
?>
```

Fig. 3. PHP source code receives the input department value from the browser and displays all staff names and salaries if the input department is found in the DB.


Fig. 4. The results displayed on the browser when the input department is "HR". The "keyword" and its associated input value "HR" are displayed in the URL.


Fig. 5. No record is found when the input department is "XY".


Fig. 6. The PHP program should show error messages if "keyword" is not used, or if there are no input values passed through GET method.

If the PHP program check.php is run without using the GET method with "keyword" and an input value, the program should terminate immediately and display an error message "No keyword is entered!", as shown in Fig. 6. This way, the web server will not waste any resources, as aforementioned. Since the web application can be accessed by many people, it can be costly if preventative measures are not taken. It is very important to teach students how to protect the servers in the client-server architecture.

The ALTER command can change the table structure; for instance, it can add or remove a column, or change the datatype and constraints. The DROP command will remove the entire table or any object from the DB. The commands CREATE, ALTER, and DROP are called Data Definition Languages (DDL) because their effects can be seen at the structure level.

### D. QUIZ 1 and Project 1

Students should take Quiz 1 (preferably 60 minutes long) which should cover the first 3 weeks of material, including homework 1 and project 1. Instructor should review the answers and discuss the homework and project assignments. Project 1 should require the students to create their first DB, complete with at least two components: 1) the ability to login with a username and password, and 2) the ability to retrieve information correctly. Note that this is a basic framework for a project, and details and implementation should be up to the instructor.

1) Retrieve all data from a "User" table provided by the instructor and display the results on the browser, as shown in Fig. 7.

The users in the database:

| ID | login ID | password | Name | Role | address | Zipcode | State |
|---|---|---|---|---|---|---|---|
| 1 | test | test | CPS3740 Tester | tester | 1000 Morris Ave. | 07083 | NJ |
| 2 | tiger | test | Huang Austin | teacher | 1000 Morris Ave. | 07083 | NJ |
| 3 | panda | test | Smith Timothy | student | 200 Union Ave. | 07101 | CA |
| 7 | lion | test | AAA XYZ | staff | 33 James St | 07331 | NJ |

Fig. 7. Display all the users and their columns from the DB.

2) A sample login page is shown in Fig. 8, and lets people enter a login ID and password. The program will verify the login ID and password with the information in the Users table in the DB.

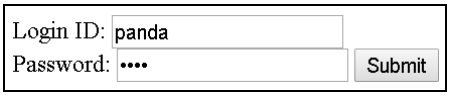



Fig. 8. Login page.

3) If the login ID and password match the records in the database, the program should show the user's IP and other information, as shown in Fig. 9.

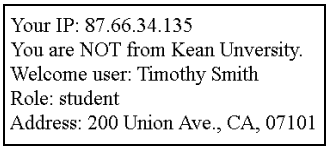



Fig. 9. User home page.

### E. SQL — Constraints; Creating Tables and Temporary Tables; Alter and DROP

Students should learn how to create simple tables in this week as well as how to use constraints - datatype, primary key, foreign key, auto increment, NOT NULL, default values, and possible range. Please note that it is very important to educate students on the impact of the data size. One extra byte of a column could result in an additional 1GB if there are 1 billion records. Therefore, students should properly choose the data types; i.e. picking between char and varchar types, or picking between ints and strings.

1) The following SQL statement details how to create a table named "Test" with 2 columns (id, name), where id is of int type and name is of varchar type. Note that id will be set as the primary key and the value will auto-increment.

*CREATE TABLE Test (id int primary key auto_increment, name varchar(50));*

2) The *auto_increment* rule will let DBMS automatically increment the value of the "id" column. The value is

indexed from 1 and is increased sequentially. It is good to use *auto_increment* when an attribute is required to have unique and sequential values, i.e. when identifying the objects, such as product ID, student ID, etc. When designing the web interface for the user to enter data for the columns, the interface should not allow the user to manually add the new value of the *auto_increment* field.

3) A temporary table can be created during the connection. Once the user logs out or the connection is closed, the temporary table will be released from memory. Since SQL does not have arrays, temporary tables can be used like 2-D arrays to store the intermediate results generated by some queries. The following SQL statement will create a temporary table named Tmp.

*CREATE TEMPORARY TABLE Tmp(id int, name varchar(100));*

4) The following SQL statement adds a new column "price" before the name column (and after id) in the Test table.

*ALTER TABLE Test add column price float after id;*

5) The following SQL statement removes the entire table from the DB.

*DROP TABLE Test;*

### F. SQL — Join and Set Operations

In a properly designed DB, different information should be kept in different tables, so tables won't have irrelevant data. The address, city, state, and other information of where a staff member is working at is stored in a separate table named "Department", as shown in Table II. The reason why the data should be separated into two tables will be discussed in the normalization section later. In order to find the address that the staff member "Tony" is working at, it should be required to first use his name to get his "dept" code "IT" in the Staff table. Then, one should use the "IT" code to retrieve his address from the "address" column in the "Department" table. The process of getting information from two or more tables is called "join"ing.

There two kinds of "join" – inner join (also called INTERSECT), which displays the common values between tables, and outer join, which shows records in either table. Often, one will use the "join" operation with a common column to associate two tables at once. Without the common column condition, all combinations of two tables will be displayed. This situation is called "Cartesian product".

TABLE II: DEPARTMENT TABLE

| dept | address | city | state |
|---|---|---|---|
| HR | 12 Main Ave. | Union | NJ |
| IT | 100 Broad St. | New York | NY |
| PJ | 28 Union Ave. | Boston | MA |
| SL | 66 Central Ave. | Houston | TX |

1) The following statement shows all combinations of joining the Staff and Department tables, which has 24 rows (6 rows from Staff x 4 rows from Department) and 10 columns (6 columns from Staff + 4 columns from Department).

*SELECT * FROM Staff, Department;*

2) The following statement uses an inner join to display the columns in the Staff and Department tables that contain common values in the "dept" column, a total of 6 records (with 10 columns).

*SELECT * FROM Staff s, Department d WHERE d.dept=s.dept;*

3) Since the column name "dept" is the same in both Staff and Department tables, the statement in 2.7.2 statement can use a "NATURAL JOIN", which will join the two tables but remove the duplicated column, "dept". This results in only 9 columns.

*SELECT * FROM Staff s NATURAL JOIN Department d;*

4) The following statement shows the address, city, and state where the staff member "Tony" is working at.

*SELECT s.name, d.address, d.city, d.state FROM Staff s, Department d WHERE d.dept=s.dept and s.name='Tony';*

5) The following statement uses LEFT OUTER JOIN to list each department, along with the staff members working in each department. If the department has no staff, the columns that correspond to staff members will be filled with NULL values, as shown in Fig. 9.

*SELECT * FROM Department d LEFT JOIN Staff s ON s.dept=d.dept;*

| dept | address | city | state | id | name | dept | sex | salary | ext |
|---|---|---|---|---|---|---|---|---|---|
| IT | 100 Broad St. | New York | NY | 101 | Mary | IT | F | 45000 | 121 |
| HR | 12 Main Ave. | Union | NJ | 102 | Smith | HR | M | 37000 | *NULL* |
| IT | 100 Broad St. | New York | NY | 103 | Tony | IT | M | 38000 | 115 |
| PJ | 28 Union Ave. | Boston | MA | 104 | Sarah | PJ | F | 35000 | 311 |
| IT | 100 Broad St. | New York | NY | 105 | Mark | IT | M | 45000 | 153 |
| HR | 12 Main Ave. | Union | NJ | 106 | Alice | HR | F | 39000 | 433 |
| SL | 66 Central Ave. | Houston | TX | *NULL* | *NULL* | *NULL* | *NULL* | *NULL* | *NULL* |

Fig. 9. The output of the LEFT JOIN query in 2.6.5.

The LEFT JOIN will force all records in the table on the left side (e.g. for the sample, this is the Department table) to be shown in the output. There is also RIGHT OUTER JOIN - all records in the table on the right side will be displayed, and FULL JOIIN – all records in both tables will be displayed.

### G. SQL — View, Aggregate Functions, and Null

In this week, students should learn about view, understand what an updatable view is, and know how to create views and updatable views. The instructor should cover how to use the 5 basic aggregate functions – COUNT, AVG, SUM, MIN, MAX, how to handle NULL values, and how to use the following set operations – INTERSECT, UNION and DIFFERENCE. If a column value is unknown, it should be stored as NULL and should not be an empty string or '0'. Instructors should explain and show that NULL values will impact the results of aggregate functions. The following examples are based on Table I.

1) The following SQL statement shows the number of female staff. Please note that the output header will also contain the aggregate function, COUNT, i.e. "COUNT(sex)", and NOT just "sex". This is shown in Fig. 10 below.

*SELECT COUNT(sex) FROM Staff WHERE sex='F';*

| COUNT(sex) |
|---|
| 3 |

Fig. 10. Result of 2.7.1.

2) When the web DB application needs to retrieve values from an aggregate function, it is required to rename the output header. The following PHP statement will change the output header from "COUNT(sex)" to "ct" in the

SQL query. See the output in Fig. 11 below.

$sql="SELECT COUNT(sex) AS ct FROM Staff WHERE sex='F'";


Fig. 11. Result of 2.7.2.

After the name is changed, the following PHP statement can use the meaningful key "ct" to retreive the value returned from the DB.

*$myCt=$row['ct'];*

3) The following SQL statement shows the maximum salary of IT department.

*SELECT MAX(salary) FROM Staff where dept = 'IT';*

4) The following SQL statement shows the number of people that have an extension number. An instructor should discuss how to use IS and IS NOT to find or exclude the NULL values.

SELECT count(ext) FROM Staff;

The statement can also be written as:

*SELECT COUNT(SID) FROM Staff WHERE ext IS NOT NULL.*

5) The following SQL statement shows how to use the MAX, COUNT, SUM and AVG functions; remember that they are impacted by the presence of NULL values.

*SELECT COUNT(salary), SUM(salary), AVG(salary), MAX(salary) from Staff;*

6) The following SQL statement shows the logic operations AND and OR with NULL values. Please note that TRUE is represented by 1 and FALSE is represented by 0 in MySQL.

*SELECT 1=1, 'a'='b', 1 AND 0, 1 OR 0, 1 AND NULL, 1 OR NULL, NULL AND NULL, NULL OR NULL*;

7) The following SQL statement will create a view *vTest* that is not updatable.

*CREATE VIEW vTest AS SELECT dept, count(*) ct FROM Staff GROUP BY dept;*

### *H. Midterm and Assignments*

The 75-minute midterm exam should cover topics from week 1 to 7. Instructor should also review the answers following the exam so students know and understand the mistakes they made. Assignments Homework 1 and Project 1 should have been due in the end of the 6th and 7th week, respectively. The instructor should provide general feedback for the students, so they know how to improve for the future. Homework 2 and Project 2 should be given to the students this week, and they are should be due in the 12th and 14th week, respectively. Homework 2 mainly covers SQL joins and stored routines.

Project 2 focuses on the integrating the learning in a practical environment, with the following requirements (2.8.1 to 2.8.4). The "search product" function should be available to anyone without login. For a user whole role is "Staff", they should be able to "Add" and "Update" product pages.

1) Create a "Product" table (id, name, description, sell_price, cost, quantity, *user_id*, *vendor_id*) where *user_id* is a foreign key that references the primary key, *id*, in the "User" table. *Vendor_id* is also a foreign key that references the primary key *id* in the "Vendor" table. Both the User and Vendor tables should be provided by the instructor.
2) 2.8.2 Implement a function to search for products, as shown in Fig. 12. The keyword should be pattern matched against the product name and description. If no product is found, a proper message should be displayed.

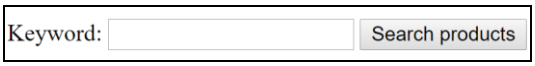



Fig. 12. A user can enter a keyword to pattern match products by name or description in the Student's "Products" table.

3) Create an add product page, as shown in Fig. 13. The program should require that all data input fields not be empty and for the numeric fields to not be negative (as well as within a reasonable price range). The sell_price must also be greater than the cost. These errors should be caught at the front-end, by JavaScript, to reduce the web server's load. If the same product already exists in the table, an error message should be displayed. If there are any errors about the data, the record should not be inserted into the DB. The vendor id and name should be retrieved from the Vendor table and only the name should be displayed in the dropdown list on the web page.

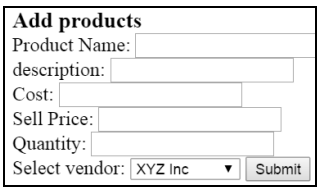



Fig. 13. An example implementation of 2.8.3.

4) Create an update product page, as shown in Fig. 14. The columns highlighted in yellow are not updatable. The Add Product checking rules (outlined in 2.8.3) should be applied to the 4 updatable columns. Staff members should be able to update multiple records at once. This means that the instructor should teach students how to pass and receive <INPUT> in an array format, though CGI. The following shows how one can use HTML to pass the product_id in an array format with the value 3 and in hidden type.

*<input type='hidden' name='product_id[]' value='3'>*

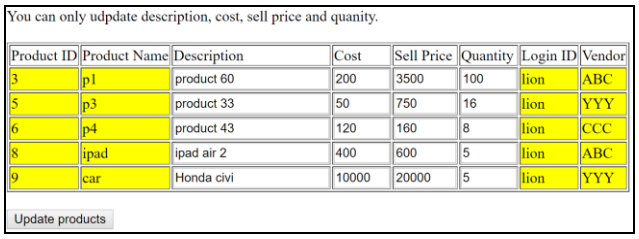
The following PHP code shows how to calculate the number of products passed from the browser, and how to receive the product_id in an array format.

```
for($i=0;$i<count($_POST['product_id']);$i++) {
  $product_id[$i]=$_POST['product_id'][$i];
}
```

### *I. SQL — Group by, Having, Insert, Update, Delete, Subquery, and in*

This week should cover how to group datasets using GROUP BY, add new records using INSERT, change data content using UPDATE, and remove a record using DELTE. From experience, grouping is one of the most challenging topics in SQL. The commands – SELECT, INSERT, UPDATE, DELETE are called Data Manipulation Language

(DML), since their effects can be seen on the data level.


You can only udpdate description, cost, sell price and quanity.

| Product ID | Product Name | Description | Cost | Sell Price | Quantity | Login ID | Vendor |
|---|---|---|---|---|---|---|---|
| 3 | p1 | product 60 | 200 | 3500 | 100 | lion | ABC |
| 5 | p3 | product 33 | 50 | 750 | 16 | lion | YYY |
| 6 | p4 | product 43 | 120 | 160 | 8 | lion | CCC |
| 8 | ipad | ipad air 2 | 400 | 600 | 5 | lion | ABC |
| 9 | car | Honda civi | 10000 | 20000 | 5 | lion | YYY |

Update products

Fig. 14. An implementation of 2.8.4.

Students are often confused between DELETE and DROP, and UPDATE and ALTER. It is very important that instructors ask students to do more exercises in class so they are able to grasp the differences between the two. The following examples are based on Table 1.

1) The following SQL statement shows the number of staff in each department.

*SELECT dept, COUNT(dept) ct FROM Staff GROUP BY dept;*

2) The following SQL statement shows the number of staff in each department with at least 1 female staff member.

*SELECT dept, sex, COUNT(dept) ct FROM Staff where sex='M' GROUP BY dept, sex having ct >=2 ;*

Based on Table I, the results of the previous two queries are shown in Fig. 15 and 16, respectively.

| dept | ct |
|---|---|
| HR | 2 |
| IT | 3 |
| PJ | 1 |

Fig. 15. Result of 2.9.1.

| dept | sex | ct |
|---|---|---|
| IT | M | 2 |

Fig. 16. Result of 2.9.2.

3) The following SQL statement shows how to increase "Sarah"'s by $1000.

UPDATE Staff SET salary=salary+1000 where name='Sarah';

4) The following SQL statement shows how to add a new record with id=107, name='Will', dept='PJ', sex='F', salary=37000, and ext is unknown.

*INSERT INTO Staff (id,name,dept,sex,salary,ext) VAUES (1, 'Will', 'PJ', 'F', 37000, NULL);*

Since all column values are provided, the SQL can also be:

*INSERT INTO Staff VALUES (1, 'Will', 'PJ', 'F', 37000, NULL);*

5) The following SQL statement shows how to delete the record added previously.

*DELETE FROM Staff WHERE name='Will';*

Instructors should emphasize that all records will be updated with the same values, or all records will be removed from the DB, if the WHERE condition is not provided.

Since the INSERT, UPDATE, DELETE commands will not return any results, it is very important to know if the command was run successfully in the PHP program. The function *mysqli_affected_rows()* should be used to detect the number rows affected.

6) A subquery is a query (inner query) inside another query (outer query). DBMS will need to execute the inner query first to get the results for the outer query to use. The following SQL statement uses subqueries to find the person's name who has the maximum salary in "IT" department. The result is shown in Fig. 17, based on the data in Table 1.

*SELECT name, salary FROM Staff WHERE salary=(SELECT MAX(salary) FROM Staff WHERE dept='IT');*

| name | salary |
|---|---|
| Mary | 45000 |
| Mark | 45000 |

Fig. 17. The correct result of 2.9.6.

Please note that the following SQL query will produce the wrong result, as shown in Fig. 18.

*SELECT name, MAX(salary) FROM Staff WHERE dept='IT';*

| name | MAX(salary) |
|---|---|
| Mary | 45000 |

Fig. 18. The wrong result of the incorrect SQL query in 2.9.6.

The DBMS will find the correct maximum salary in the "IT" department. Since MAX is an aggregate function that produces a single value from populating several records, DBMS is not able to associate the name with the MAX function in one step. Therefore, it will pick the first name it finds in the table that matches the maximum salary. It is necessary to use a subquery to find the maximum salary of the 'IT' department, and then output all names that have this maximum salary value. This is because it is possible to have more than 1 maximum salary, as shown in Fig. 17.

### J. SQL — Variables and Stored Routines

So far, students have learned SQL by using singular statements. This is not real programming. Like learning any computer languages, it is very important to learn how to write procedures and functions. In SQL, they are called stored procedures and stored functions because programs will be stored on a DB server after DBMS compiles the statements (if there are no syntax errors). In MySQL, stored routines is a general term that refers to both routines and procedures.

Before students can write a stored routine, they have to learn about 3 types of SQL variables – local, session, and global. Students also must learn how to assign a value (constant or variable) to a variable. There are 3 types of arguments for a procedure – IN, OUT, and INOUT – but functions only have an IN type. In addition, instructors should cover the SQL functions CONCAT() and GROUP_CONCAT(), and how to use LIKE for pattern matching an input argument in stored routines.

### K. Security, Grant and Revoke

Web DB security can be classified into 4 the following types:

1) **DB user and privilege management.** This has to be done by the following SQL commands: GRANT, which can give particular privileges to specific objects in the DB (i.e. also control which users can access the

privileges from which hosts), and REVOKE, which can remove the privileges. The following SQL statement gives the SELECT privilege on the Staff table in the Test database to a user named 'tester', who can access the DB from any host (%).

*GRANT SELECT ON Test.Staff to tester@% with grant option;*

2) **Web page authentication (e.g. login).** This often requires the DB designer to store user login and password information in a table, for example, the "User" table in the class project. The login will use the POST method to pass the login and password to the middle-end program, which will verify with the data in the DB. The password should be encrypted by SHA256, or a better hash function. The password should not be readable directly select from the table. For example, Fig. 7 is a poor implementation of the password.
3) **DB web page authorization through role control.** Another way to improve web DB application security is to use roles to control who can access what web pages or functions. The role can be a numeric level or a string, i.e. "staff", "student" shown in Fig. 7. The search function shown in Fig. 12 can be accessed by anyone, but the add product page shown in Fig. 13 and update product page shown in Fig. 14 should be only accessed by "staff".
4) **Using cookies and sessions to control when the web page** should **expire.** Authentication cookies are the most common method used by web servers to know whether the user is logged in or not, and which account they are logged in with [13].

### L. QUIZ 2 and Project 2

The 60-minute Quiz 2 should cover materials from week 9 to 11, and the answers should be reviewed. Instructors should use this week to check the status of project 2 and help students improve their web DB programming skills.

### M. Planing and Requiremens

It is better to discuss DB planning and requirements after students have completed their first web DB application assignment, which will be discussed in section 3. This way, they already have hands-on experience and know about the DB features, function, and GUI firsthand and understand the importance of these requirements. In order to mimic the real world, instructors could have a group take-home exam after students learn all the basic SQL queries and let students play different roles (system engineer, database designer, software developer, and business staff) to plan a design for a mini-online store and write the requirements. This way, students will have the opportunity to brainstorm together and work as a team. Teamwork is very critical in web DB applications.

### N. Logic Design and E-R Model

Traditional DB textbooks often introduce logic design and the E-R model early in the course. However, from teaching experiences, many students won't be able to fully understand the real meaning of data modeling before they have any hands-on experience with DB. Therefore, this paper proposes to move the logic design and E-R model to later parts of the curriculum. In addition to learning how to model the data from the requirements, students should also learn how to convert E-R diagrams to schema diagrams.

### O. Normalization and Physical Design

For normalization, the instructor should cover traditional decomposition, functional dependency, loss-less join, and 1st, 2nd and 3rd normal forms. The Boyce–Codd normal form can be optional. Students should be able to break a big table into several smaller tables and set the primary keys and foreign keys to link the smaller tables together.

Physical designs include creating the following objects in the DB: project database instances, user accounts with proper privileges, stored routines, and tables based on the schema diagrams with the necessary columns to be indexed. Instructors should also teach how to estimate physical disk sizes for all the tables and recommend the partition size that will store the database files. It is also important to show students how to check where the database files are located on the server. For example, the following SQL statement shows the data folder for MySQL:

*SHOW VARIABLES LIKE 'datadir';*

Physical design should not only cover database design, but should also include application design. Since project 2 is due at the end of 14th week, the instructor should give overall feedback about students' projects and homework, especially how the implementation relates to physical design – data type and validation, data flow between the 3-tier architecture, the relationship between tables, and date storage locations on the DB server.

### P. Final Exam

The exam should have two components – a take-home database design and in-class individual written exam. The take-home is a teamwork assignment that requires a team to have 5-8 students and design a mini database application. Each team is given two forms – one for registration and one for a report. The samples are shown in Fig. 19 and 20, respectively. Each team should have 4 different roles – business staff, system engineer, database designer, and software developer.

The business staff should write several business rules based on the two forms. The following are examples based on the two forms:

1) The registration should take 4 data fields.
2) The product information should not be shown in the URL when they are passed to backend.
3) The Graphic User Interface is based on the browser.
4) The ID should be automatically assigned by the system.

The system engineer should write the detailed requirements based on the business rules and the forms.

The database designer should draw the E-R diagram, create table statements, and perform any tasks related the database.

The software developer should write the detailed code for both front-end and back-end. The code should include the following:

1) HTML code to get user input and button to submit the input to the webserver.
2) PHP code to validate the input data ranges before sending them to the database server.

3) PHP code to send the query to the database for inserting and retrieving records.
4) PHP code to display the results returned from the database on the browser.

The in-class exam should include all topics covered in the semester, and be weighted 30% for topics covered before midterm and 70% after the midterm.

**Product registration form**
ID: assigned by computer
Name: _____ (required)
Price: _____ (required)
Quantity: _____ (required)

Fig. 19. The form to register a product.

**Product Report**

| ID | Name | Price | Quantity |
|---|---|---|---|
| 1 | Camera | 149.99 | 5 |
| 2 | Desk | 19.99 | 8 |
| 3 | TV | 299.99 | 4 |

Fig. 20. The report to list the products with id, name and price.

## III. Student Learning Outcome (SLO)

Upon completion of this course, the student will be able to:

1) Explain and write basic SQL statements using the SELECT, INSERT, UPDATE, DELETE commands.
2) Explain and write basic SQL statements to create and drop tables, views, and stored routines.
3) Explain and write SQL statements to join multiple tables with conditions, and group and sort the results.
4) Design and build web database applications with authentication to retrieve, display, and update data using the PHP language.
5) Explain and demonstrate the normalization concepts and process.

## IV. Conclusions

This paper proposes a fundamental curriculum for an undergraduate database course. The main goal of this proposal is to focus on hands-on experience. Students will need to write a two-phase and 3-tier comprehensive project that integrates the web browser, web server, and database management system. Several software programs were proposed to help students quickly complete the project.

## References


[1] C. Coronel and S. Morris, *Database Systems: Design, Implementation, & Management*, 13th Edition, 2018.
[2] P. DuBois, *MySQL*, 5 Edition, Addison-Wesley Professional, April 12, 2013.
[3] L. Ullman, *PHP for the Web: Visual QuickStart Guide*, 5th Edition, Peachpit Press, July 25, 2016.
[4] A. Peicevic, *Apache HTTP Server Introduction*, 2nd Edition, CreateSpace Independent Publishing Platform, February 9, 2017.
[5] L. Welling and L. Thomson, *PHP and MySQL Web Development*, 5th Edition, Addison-Wesley Professional, September 30, 2016.
[6] XAMPP Apache + MariaDB + PHP + Perl. (2018). *Apache Friends.* [Online]. Available: https://www.apachefriends.org/
[7] M. Delisle, *Mastering phpMyAdmin 3.4 for Effective MySQL Management*, Packt Publishing, February 7, 2012.
[8] J. Eckstein and B. R. Schultz, *Introductory Relational Database Design for Business, with Microsoft Access*, Wiley, January 16, 2018.
[9] G. Gupta, *Mastering HTML5 Forms*, Packt Publishing, November 22, 2013.
[10] PHP 5 Tutorial. (2018). [Online]. Available: https://www.w3schools.com/php/
[11] M. Wright, *How To Setup A Linux Web Server*, CreateSpace Independent Publishing Platform, January 16, 2014.
[12] L. Ullman, *PHP and MySQL for Dynamic Web Sites*, 5th Edition, Peachpit Press, November 13, 2017.
[13] J. LeBlanc and T. Messerschmidt, *Identity and Data Security for Web Development: Best Practices*, O'Reilly Media, June 20, 2016.



**Ching-Yu Huang** is an assistant professor of the School of Computer Science at Kean University, Union, New Jersey, USA since September 2014. Dr. Huang received a Ph.D. in computer & information science from New Jersey Institute of Technology, Newark, New Jersey, USA.

Prior to joining Kean University, Dr. Huang had more than 16 years of experience in the industry and academics in software development and R&D in bioinformatics. His research focuses SNP genotype calling and cluster detection; image processing and pattern recognition, especially in microarray and fingerprint; geotagged images and location information reconstruction; database application development; data processing automation; E-learning, educational multimedia, methodology, and online tools for secondary schools and colleges. Dr. Huang has more than 40 publications in journals and conferences and more than 20 presentations in workshops and invited lectures.